The Coin

Uma moeda digital de prova de trabalho, leve e democrática, com contratos de pagamento e governança on-chain

Whitepaper — versão 0.1

Setembro de 2026 · the-coin.cloud

Resumo. The Coin (TCN) é uma rede monetária aberta e ponto a ponto, projetada para que máquinas modestas — um VPS de 1 vCPU e 1 GB de RAM — possam validar a cadeia completa e competir de forma justa pelas recompensas de mineração. A prova de trabalho **CoinHash** usa Argon2id com 16 MiB de memória, reduzindo a vantagem de GPUs e ASICs. A oferta é limitada a 50 milhões de TCN, emitidos por halvings ao estilo Bitcoin, com 93,75% da oferta distribuída num ciclo de estabilização de aproximadamente cinco anos, sem pré-mineração. O modelo de contas com compromisso de estado homomórfico (LtHash) mantém a validação barata e os dados compactos. A versão 0.1 inclui transferências, pagamentos em lote, cinco modelos de contratos de pagamento (escrow, vesting, assinatura recorrente, HTLC e multisig) e uma governança bicameral na qual detentores e mineradores precisam concordar para alterar parâmetros da rede. Este documento descreve o que foi implementado, como a rede funciona e por que cada decisão foi tomada.

Sumário

1 Introdução

O Bitcoin demonstrou que é possível manter um livro-razão público, resistente à censura e à falsificação, sem autoridade central. Quinze anos depois, porém, a mineração tornou-se uma indústria de ASICs especializados e energia barata, e rodar um nó completo exige recursos que afastam o usuário comum. Redes com contratos inteligentes, por sua vez, frequentemente trocaram a descentralização por desempenho, exigindo servidores potentes para validar a cadeia.

The Coin parte de uma pergunta simples: **e se a unidade mínima de participação na rede fosse um VPS barato?** Milhões de servidores virtuais de baixo custo existem no mundo inteiro. Se cada um deles puder validar blocos, retransmitir transações e minerar com chances proporcionais ao seu trabalho, a rede se torna mais distribuída geograficamente, mais difícil de capturar e mais democrática.

Os objetivos da versão 0.1 são:

- **Leveza:** validar a cadeia completa com 1 vCPU, 1 GB de RAM e poucos GB de disco.
- **Segurança:** regras de consenso simples, determinísticas e auditáveis; nenhuma transação confirmada pode ser alterada sem refazer o trabalho acumulado.
- **Mineração justa:** prova de trabalho **memory-hard** amigável a CPUs comuns.
- **Política monetária previsível:** 50 milhões de TCN, sem pré-mineração, emissão por halvings.
- **Utilidade:** pagamentos, cobranças, contratos de pagamento e troca atômica com outras redes.
- **Evolução democrática:** mudanças de parâmetros aprovadas on-chain por detentores **e** mineradores.
- **Abertura:** especificação, API e padrões de carteira documentados para que qualquer pessoa crie carteiras, exploradores e implementações alternativas.

> **Status.** Este documento descreve a implementação de referência v0.1 em Rust (repositório `thecoin`). Onde houver divergência, o código e a especificação `docs/PROTOCOL.md` são normativos.

2 Visão geral do sistema

A rede é formada por nós idênticos (`thecoind`). Cada nó mantém a cadeia completa de blocos, valida cada regra de consenso de forma independente, retransmite blocos e transações pela rede ponto a ponto e, opcionalmente, minera. Carteiras (`thecoin-wallet` ou de terceiros) guardam as chaves privadas localmente e só enviam transações já assinadas a um nó.

Componente	Responsabilidade
`thecoin-core`	Regras de consenso puras: tipos, codificação canônica, criptografia, CoinHash, LWMA, emissão, contratos, governança e função de transição de estado. Sem I/O.
`thecoin-storage`	Banco de dados embarcado redb (ACID), blocos e dados de undo comprimidos com zstd, índices de transações e endereços.
`thecoin-node`	Gerenciador da cadeia (validação, escolha do fork, reorganizações, poda), mempool, rede P2P, minerador de CPU e API REST.
`thecoin-wallet`	Chaves HD (BIP-39/SLIP-0010), arquivo de carteira criptografado, construção e assinatura de transações, URIs de pagamento, CLI.
Instalador	Script que instala binários verificados, cria carteira, configura serviço systemd endurecido e inicia a mineração.
Site / explorador	the-coin.cloud: estatísticas ao vivo, explorador de blocos, painel de governança e downloads, alimentado pela API dos nós.

Tabela 1: Componentes da implementação de referência.

O fluxo básico é o mesmo do Bitcoin: usuários assinam transações; nós as validam e as guardam no **mempool**; mineradores agrupam transações em blocos e procuram um **nonce** que satisfaça a prova de trabalho; o bloco encontrado é propagado; cada nó o valida integralmente e, se ele estender a cadeia com mais trabalho acumulado, o adota.

3 Modelo de dados

3.1 Unidades

A unidade de conta é o **TCN**. A menor fração é o **mote**: 1 TCN = 100 000 000 motes. Todos os valores são inteiros de 64 bits sem sinal; ponto flutuante nunca é usado no consenso. A oferta máxima (5×10^{15} motes) cabe com folga em 64 bits e também é um inteiro seguro em JavaScript, o que simplifica carteiras web.

3.2 Codificação canônica e hashes

Todas as estruturas de consenso são serializadas com **Borsh**: inteiros little-endian de tamanho fixo, vetores e strings prefixados por comprimento `u32`, enums com tag `u8`. Cada objeto tem exatamente uma codificação válida, o que impede ambiguidades na assinatura e no hash.

Todo hash do protocolo usa BLAKE3 com separação de domínio:

$$H_{\text{tag}(x)} = \text{BLAKE3}(\text{"TheCoin:"} \,\|\, \text{tag} \,\|\, 0\text{x}00 \,\|\, x)$$

Tags distintas (`txid`, `block`, `tx-sign`, `merkle-leaf`, `address`, …) garantem que um valor calculado para um propósito jamais colida com outro.

3.3 Contas e endereços

The Coin usa o **modelo de contas**: cada endereço possui saldo e **nonce** (número de transações já enviadas). Comparado ao modelo UTXO, o estado é menor, a validação de contratos é direta e carteiras leves precisam consultar um único registro.

O endereço é formado pelos 20 primeiros bytes de H_{address}(chave pública Ed25519) e representado em **bech32m** (BIP-350), com prefixo por rede: `tc1…` (mainnet), `tct1…` (testnet) e `tcr1…` (regtest). O checksum cobre o prefixo, então um endereço de outra rede ou com erro de digitação é rejeitado.

3.4 Transações

Campo	Tipo	Descrição
`version`	u8	Versão do formato (1).
`chain_id`	u32	Identificador da rede — impede reutilizar uma transação em outra rede.
`nonce`	u64	Deve ser igual ao nonce atual da conta; impede replay.
`fee`	u64	Taxa em motes, no mínimo `min_fee_per_byte × tamanho`.
`expiry_height`	u64	Última altura em que pode ser incluída (0 = sem validade).
`action`	enum	Transferência, lote, criar/chamar contrato, propor, votar.
`public_key`	32 bytes	Chave pública Ed25519 do remetente.
`signature`	64 bytes	Assinatura Ed25519 sobre $H_{\text{tx-sign}}$(body).

Tabela 2: Estrutura de uma transação.

O identificador é txid = H_{txid}(transação serializada). A verificação Ed25519 é **estrita** (RFC 8032, rejeitando chaves de ordem pequena e assinaturas não canônicas), eliminando a maleabilidade: o txid só muda se o dono da chave assinar outra transação.

Uma propriedade útil para carteiras: como `fee` tem largura fixa, o tamanho da transação não depende do valor da taxa. Basta assinar uma vez com taxa zero para descobrir o tamanho exato e então assinar com a taxa final. Uma transferência simples ocupa cerca de 160 bytes.

Uma transação é **atômica**: ou é totalmente válida e aplicada, ou não pode entrar em um bloco. Não existe o estado "falhou mas cobrou taxa", o que torna o comportamento previsível para usuários.

3.5 Blocos

Campo	Tamanho	Descrição
`version`	4	Versão do cabeçalho (1).
`height`	8	Altura do bloco (gênese = 0).
`prev_hash`	32	Hash do cabeçalho anterior.
`tx_root`	32	Raiz Merkle dos txids.
`state_root`	32	Compromisso do estado global **após** aplicar o bloco (LtHash).
`timestamp`	8	Horário Unix em segundos.
`target`	32	Alvo da prova de trabalho (inteiro de 256 bits big-endian).
`nonce`	8	Campo livre para a busca da prova de trabalho.
`miner`	20	Endereço que recebe recompensa e taxas.
`signal`	4	Bits de sinalização de governança dos mineradores.

Tabela 3: Cabeçalho de bloco: tamanho fixo de 180 bytes.

O identificador do bloco é H_{block}(cabeçalho), barato de calcular. A prova de trabalho é um hash separado e caro (CoinHash). Não existe transação **coinbase**: a recompensa é definida pelo campo `miner` e registrada no estado.

A árvore Merkle usa folhas e nós com tags distintas e promove o último elemento de níveis ímpares sem duplicá-lo, eliminando a ambiguidade conhecida do Bitcoin (CVE-2012-2459).

4 Consenso por prova de trabalho

4.1 CoinHash

$$\text{CoinHash}(\text{cabeçalho}) = \text{Argon2id}(\text{senha} = \text{cabeçalho}, \text{sal} = \text{"TheCoin/PoW/v1.."}, m = 16\ \text{MiB}, t = 1, p = 1)$$

O resultado de 32 bytes, lido como inteiro big-endian, deve ser menor ou igual ao `target` do cabeçalho. Argon2id (RFC 9106) é o vencedor da Password Hashing Competition e é **memory-hard**: cada tentativa exige preencher 16 MiB de memória em ordem dependente dos dados. As consequências são:

- **CPUs comuns são competitivas.** O gargalo é a largura de banda de memória, não o número de transistores de lógica; GPUs e ASICs têm vantagem muito menor do que em SHA-256.
- **Verificação barata para nós modestos.** Um hash custa milissegundos, e cada nó verifica apenas um hash por bloco.
- **Memória previsível.** Cada thread de mineração usa 16 MiB; um VPS de 1 GB minera sem pressão.

Memória Argon2id	Tempo por hash	Hashes/s por núcleo
4 MiB	2,41 ms	415
8 MiB	5,22 ms	192
16 MiB (escolhido)	**12,04 ms**	**83**
32 MiB	25,16 ms	40

Tabela 4: Medições em um VPS OVH de 2 vCPU (Haswell). 16 MiB equilibra resistência a hardware especializado e custo de verificação.

Com 83 hashes/s, verificar a prova de trabalho de um ano inteiro de blocos (525 960 blocos) custa cerca de 1,75 hora de CPU num único núcleo; o nó verifica blocos recebidos em paralelo em todos os núcleos durante a sincronização, mantendo a ordem de aplicação.

4.2 Ajuste de dificuldade (LWMA-1)

Redes jovens de máquinas pequenas sofrem grandes variações de **hashrate** à medida que mineradores entram e saem. The Coin recalcula o alvo a cada bloco com a média móvel linearmente ponderada LWMA-1 (Zawy), com janela $N = 60$ blocos e tempo-alvo $T = 60$ s:

$$t_j = \min\big(6T, \max\big(s_j, s_{j-1} + 1\big) - s_{j-1}\big)$$

$$\text{alvo}_{n+1} = \frac{\sum_{j=1}^{N} \text{alvo}_j}{N} \cdot \frac{\sum_{j=1}^{N} j \cdot t_j}{k}, \quad k = N(N+1)\frac{T}{2}$$

Blocos recentes pesam mais, então a dificuldade reage em poucos blocos, enquanto os limites em t_j neutralizam timestamps manipulados. O cálculo usa aritmética inteira de 512 bits e o alvo é limitado pelo **pow limit** da rede. Os primeiros $N + 1$ blocos usam o alvo da gênese.

4.3 Regras de tempo

- O timestamp deve ser maior que a mediana dos 11 blocos anteriores (**median time past**).
- O timestamp não pode estar mais de 180 s à frente do relógio local do nó.

4.4 Escolha da cadeia e finalidade

Cada nó segue a cadeia válida com **maior trabalho acumulado** ($\sum 2^{256}/(\text{alvo}+1)$); em empate, a primeira vista. Uma reorganização desconecta blocos usando dados de **undo** e conecta os blocos do novo ramo, tudo em uma única transação atômica do banco de dados — se qualquer bloco do novo ramo for inválido, nada muda.

Como proteção adicional de uma rede em crescimento, reorganizações com mais de **720 blocos** (≈ 12 horas) são recusadas, e checkpoints podem ser fixados no código a partir de alturas já consolidadas. Isso limita o dano de um ataque de 51% de curta duração a janelas recentes.

5 Política monetária

5.1 Emissão

A emissão segue o modelo do Bitcoin, comprimido para o horizonte de estabilização desejado:

$$\text{subsídio}(h) = \begin{cases} 0 & \text{se } h = 0 \\ 40\ \text{TCN} \gg \lfloor (h-1) / \ 625\ 000 \rfloor & \text{caso contrário} \end{cases}$$

Com blocos de 60 s são produzidos ≈ 525 960 blocos por ano, e um halving ocorre a cada ≈ 434 dias (1,19 ano). A soma geométrica limita a oferta:

$$\sum \text{subsídios} < 40 \times 625\ 000 \times 2 = 50\ 000\ 000\ \text{TCN}$$

Por causa do truncamento inteiro dos halvings, o total emitido fica alguns motes abaixo de 50 milhões — o limite nunca é ultrapassado, e cada nó rejeita um bloco que faria a emissão acumulada exceder 50 milhões.

Era	Recompensa	Emitido na era	Acumulado	% da oferta	Fim (anos)
1	40 TCN	25 M	25 M	50%	1.19
2	20 TCN	12.5 M	37.5 M	75%	2.38
3	10 TCN	6.25 M	43.75 M	87.5%	3.56
4	5 TCN	3.125 M	46.875 M	93.75%	4.75
5	2.5 TCN	1.563 M	48.438 M	96.88%	5.94
6	1.25 TCN	0.781 M	49.219 M	98.44%	7.13
7	0.625 TCN	0.391 M	49.609 M	99.22%	8.32
8	0.313 TCN	0.195 M	49.805 M	99.61%	9.51

Tabela 5: Cronograma de emissão. As quatro primeiras eras formam o ciclo de estabilização (≈ 4,75 anos, 93,75% da oferta).

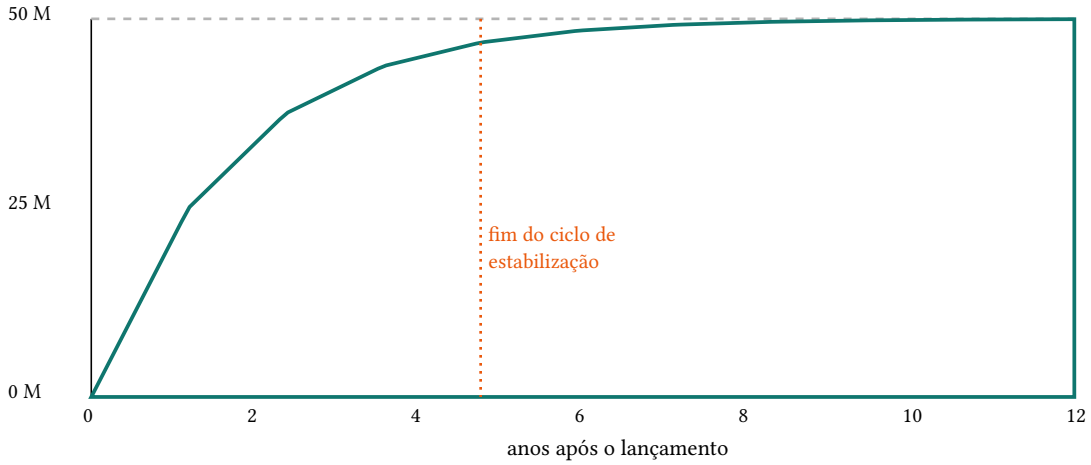

Figura 1: Oferta emitida ao longo do tempo (limite de 50 milhões de TCN).

5.2 Por que um ciclo de cinco anos

O Bitcoin leva mais de uma década para emitir a maior parte da oferta. Uma rede nova e pequena precisa atrair participantes rapidamente e, ao mesmo tempo, chegar logo a uma oferta estável que não dependa de inflação alta. Em ≈ 4,75 anos, 93,75% das moedas estão distribuídas a quem efetivamente protegeu a rede. A partir daí, a emissão continua caindo pela metade a cada ≈ 1,19 ano (≈ 1,56 milhão de TCN na quinta era) e a segurança passa gradualmente a ser financiada pelas taxas.

5.3 Maturação, taxas e queima

- A recompensa do bloco (subsídio + taxas) só se torna gastável após **100 blocos** (≈ 100 minutos), evitando que moedas de um bloco revertido já tenham sido gastas.
- Toda transação paga no mínimo `min_fee_per_byte × tamanho` (10 motes/byte na mainnet, parâmetro de governança). As taxas vão integralmente ao minerador.
- Depósitos de propostas de governança que não atingem quórum são **queimados** e contabilizados; a oferta circulante é `emitido − queimado`.
- Não há pré-mineração, alocação para fundadores ou reserva de desenvolvimento: todo TCN nasce da prova de trabalho.

6 Estado global e armazenamento compacto

6.1 Registros de estado

O estado é um mapa chave-valor com prefixos: contas (saldo, nonce, bloqueio de votos), contratos, propostas, votos, recompensas pendentes de maturação e um registro global (emissão, queima, parâmetros de governança, propostas em votação). Contas vazias não são armazenadas; contratos concluídos são apagados. O estado cresce com o número de contas ativas, não com o histórico.

6.2 Compromisso homomórfico de estado (LtHash)

Todo cabeçalho contém a raiz do estado após o bloco. Recalcular uma árvore Merkle de todo o estado a cada bloco seria caro para máquinas pequenas. The Coin usa **LtHash16**: cada registro (k, v) é expandido por BLAKE3--XOF em um vetor de 2048 inteiros de 16 bits, e o compromisso é a soma, módulo 2^{16}, dos vetores de todos os registros:

$$L(\text{estado}) = \sum_{(k,v)\in\ \text{estado}} \text{XOF}(k, v) \bmod 2^{16}, \quad \text{state\_root} = H_{\text{state-root}(L)}$$

Inserir ou remover um registro é somar ou subtrair um vetor, então atualizar o compromisso custa proporcionalmente aos registros alterados pelo bloco. A segurança se apoia no problema de reticulados **Short Integer Solution**. Qualquer divergência — bug, corrupção de disco ou manipulação — faz o nó rejeitar o bloco imediatamente, e ao iniciar o nó confere que o compromisso salvo corresponde ao topo.

6.3 Banco de dados e compressão

- **redb**: banco embarcado em Rust puro, com transações ACID e árvores B copy-on-write. Cada bloco — estado, undo, índices, topo e compromisso — é gravado em **uma** transação: uma queda de energia nunca deixa um bloco pela metade.
- **zstd**: corpos de blocos e dados de undo são comprimidos.
- **Undo limitado**: dados para reorganização são mantidos só para os últimos 736 blocos.
- **Poda opcional**: nós podados mantêm apenas os últimos N corpos de blocos (mínimo 1000), mais cabeçalhos e estado; nós arquivo mantêm tudo e servem o histórico público.
- **Cache configurável** (64 MB por padrão).

7 Transações e pagamentos

- **Transferência** com **memo** de até 256 bytes (número de pedido, fatura, mensagem).
- **Pagamento em lote** para até 128 destinatários em uma única transação (folha de pagamento, pools, exchanges).
- **Validade** opcional (`expiry_height`) para que cobranças antigas não sejam executadas tarde demais.
- **Substituição por taxa**: uma transação pendente pode ser substituída por outra de mesmo nonce com taxa ao menos 25% maior.
- **Solicitações de pagamento** padronizadas por URI, prontas para QR code:

```
thecoin:tc1q...?amount=12.5&memo=Pedido%20%23123&label=Loja
```

8 Contratos de pagamento

Em vez de uma máquina virtual de uso geral, a v0.1 oferece **modelos nativos de contratos de pagamento**. Eles não são Turing-completos, não têm gás, laços ou reentrância, e cada chamada tem custo fixo — portanto

são seguros e baratos de validar em máquinas pequenas. Os fundos ficam no registro do contrato e só saem pelas regras abaixo.

Modelo	Regras	Casos de uso
Escrow	Pagador bloqueia o valor. Pagador ou árbitro liberam ao recebedor; recebedor ou árbitro reembolsam; após o prazo o pagador reembolsa sozinho.	Comércio eletrônico, serviços, compra entre desconhecidos.
Vesting	Libera linearmente entre início e fim, nada antes do **cliff**. Se revogável, o criador recupera a parte não liberada.	Salários, bônus, distribuição a equipes.
Assinatura	Pagamento pré-pago de N períodos; o recebedor saca um período imediatamente e mais um a cada `period_blocks`; o pagador cancela e recupera os períodos futuros.	SaaS, mensalidades, aluguel.
HTLC	Paga ao destinatário quem revelar a pré-imagem com SHA-256 igual ao **hash lock** até o prazo; depois, o remetente recupera.	Trocas atômicas com Bitcoin e outras redes, canais de pagamento.
Multisig	Cofre controlado por M de N signatários (até 16); gastos propostos e aprovados on-chain.	Tesouraria de empresas e associações, custódia compartilhada.

Tabela 6: Modelos de contratos de pagamento da v0.1.

O identificador de um contrato é $H_{\text{contract-id}}(\text{criador} \,\|\, \text{nonce})$, conhecido antes mesmo da confirmação. Novos modelos — e futuramente uma VM WebAssembly com medição de custo — podem ser adicionados como novas variantes, ativadas por uma atualização aprovada pela governança.

9 Governança bicameral

The Coin foi pensada para evoluir sem donos. Mudanças de parâmetros são decididas on-chain por um sistema **bicameral**: uma proposta só passa se as duas câmaras aprovarem.

1. **Câmara dos detentores.** Qualquer endereço vota com um peso em TCN. As moedas usadas no voto ficam **bloqueadas** até o fim da votação e não podem ser movidas — portanto não podem ser transferidas para outro endereço e usadas de novo. Cada endereço vota uma vez por proposta.
2. **Câmara dos mineradores.** Cada proposta em votação recebe um bit do campo `signal`; mineradores que apoiam a proposta ligam o bit nos blocos que produzem.

Critério	Regra (em pontos-base, 10 000 = 100%)	Padrão mainnet
Quórum	votos totais (sim + não + abstenção) ≥ `quorum_bp` × oferta circulante	10%
Aprovação dos detentores	sim ≥ `approval_bp` × (sim + não)	66,67%
Aprovação dos mineradores	blocos sinalizando ≥ `miner_approval_bp` × blocos da votação	60%
Duração	`vote_period` blocos	20 160 (≈ 14 dias)
Ativação	`activation_delay` blocos após aprovação	2 880 (≈ 2 dias)
Depósito	`proposal_deposit`, devolvido se houver quórum, queimado caso contrário	100 TCN

Tabela 7: Regras de aprovação de propostas.

Ciclo de vida: `Votação` → ao fim, `Aprovada` ou `Rejeitada` → após o atraso, `Ativada`. O atraso dá tempo para operadores de nós atualizarem o software quando necessário.

O que pode mudar: tamanho máximo de bloco, taxa mínima, depósito, duração da votação, quórum, limiares de aprovação e atraso de ativação — sempre dentro de limites rígidos codificados (por exemplo, bloco entre 250 kB e 8 MB). Propostas de texto registram decisões da comunidade, e propostas de **atualização de software** registram a versão aprovada; nós desatualizados passam a exibir um aviso.

> **O que nunca pode mudar por votação:** o limite de 50 milhões de TCN, o cronograma de emissão, o algoritmo de prova de trabalho e as regras de validade de transações. Alterá-los exigiria que cada operador instalasse voluntariamente um software diferente — exatamente como no Bitcoin.

Nenhum grupo decide sozinho: grandes detentores não mudam a rede sem os mineradores, e grandes mineradores não mudam a rede sem os detentores. O site the-coin.cloud exibe cada proposta, o andamento das duas câmaras, o quórum e o tempo restante, lendo diretamente a API dos nós.

10 Rede ponto a ponto

- **Transporte:** TCP. Cada mensagem é um **frame** com magic da rede (4 bytes), tamanho, checksum e payload Borsh; frames acima de 9 MB ou malformados derrubam a conexão.
- **Handshake:** troca de `Version` (gênese, altura, serviços, nonce anti-autoconexão) e `Verack`. Nós de outra rede ou gênese são desconectados.
- **Sincronização:** localizadores de blocos (densos no topo, esparsos para trás) → até 500 hashes por `Inv` → `GetData` → blocos. A prova de trabalho dos blocos recebidos é verificada em paralelo e os blocos são aplicados em ordem por uma thread dedicada.
- **Propagação:** novos blocos e transações são anunciados por `Inv` apenas a quem ainda não os conhece; o mempool de um novo par é solicitado na conexão.
- **Descoberta:** seeds DNS (`seed1.the-coin.cloud`, `seed2.the-coin.cloud`), troca de endereços e um gerenciador de endereços persistente com **back-off** exponencial.
- **Proteção contra abuso:** limites de conexões de entrada (32) e por IP, filas de escrita limitadas (pares lentos são desconectados), pontuação de mau comportamento com banimento de 24 h, limites estruturais de mensagens, timeouts de handshake e ping.

11 Nó, mineração e instalador

11.1 Modelo de execução

O nó foi desenhado para não travar a máquina nem a si mesmo:

- rede e API usam um runtime assíncrono Tokio com apenas 2 threads;
- blocos são validados por **uma** thread dedicada, com fila, isolando trabalho pesado da rede;
- leituras (API, P2P) usam transações MVCC do banco e não bloqueiam a escrita;
- threads de mineração rodam com prioridade mínima (`nice 19`) e, por padrão, usam todos os núcleos menos um;
- a mineração pausa enquanto o nó sincroniza e reconstrói o bloco-modelo a cada novo topo ou lote de transações;
- a API REST tem limite de concorrência, de corpo e de tempo por requisição.

11.2 Instalação

```
curl -fsSL https://the-coin.cloud/install.sh | sudo bash
```

O instalador detecta a arquitetura, baixa os binários e **verifica o SHA-256**, recorre à compilação do código-fonte quando não há binário, oferece criar swap em máquinas sem swap, cria o usuário de sistema `thecoin`, cria a carteira que receberá as recompensas (a frase de recuperação é mostrada uma única vez), gera `/etc/thecoin/thecoind.toml`, instala um serviço systemd com isolamento (`ProtectSystem=strict`, `NoNewPrivileges`, sem capabilities), abre a porta P2P no firewall e inicia o nó, que imediatamente começa a sincronizar e minerar.

11.3 Requisitos

Recurso	Mínimo	Recomendado
CPU	1 vCPU x86_64 ou ARM64	2+ vCPU
Memória	1 GB (com swap)	2 GB
Disco	5 GB	20 GB SSD
Rede	porta TCP 7333 aberta	IP público estável
Sistema	Linux com systemd	Ubuntu/Debian LTS

Tabela 8: Requisitos de um nó minerador.

12 Carteira e padrões para desenvolvedores

A carteira de referência usa exclusivamente padrões abertos, para que qualquer carteira compatível derive os mesmos endereços a partir das mesmas palavras:

- **BIP-39**: frase de recuperação de 12 ou 24 palavras em inglês.
- **SLIP-0010 Ed25519**: caminho `m/44'/7333'/conta'/0'/índice'` (todos endurecidos).
- **Arquivo de carteira v1**: JSON com a frase cifrada por ChaCha20-Poly1305 com chave derivada da senha por Argon2id (64 MiB, 3 passes), gravado com permissão 0600.
- **Transações**: codificação Borsh documentada byte a byte, com vetores de teste.
- **API REST**: consulta de saldo, `next_nonce` (considerando o mempool), sugestão de taxa, envio de transação assinada, histórico, contratos e propostas.

A biblioteca `thecoin-core` não tem I/O e compila para qualquer plataforma suportada pelo Rust, inclusive WebAssembly, o que permite carteiras web e móveis reutilizarem exatamente as regras de consenso. A documentação do repositório (`docs/WALLET_DEVELOPERS.md`, `docs/PROTOCOL.md`, `docs/API.md`) descreve tudo o que é necessário para criar novas carteiras.

13 API e integração com o site

Cada nó expõe uma API REST JSON (`/api/v1`) com status da rede, oferta, blocos, transações, endereços e histórico, contratos, propostas de governança e parâmetros, mempool, pares e mineração. Valores são inteiros em motes, hashes em hexadecimal e endereços em bech32m.

A API vem ligada apenas em `127.0.0.1`. Para o site the-coin.cloud, hospedado em outra máquina, os nós seed expõem a API somente ao IP do servidor web, e o nginx do site faz **proxy** com TLS, cache curto, limite de requisições e **failover** entre os dois nós. O explorador, o painel de governança e as estatísticas da página inicial consomem essa API — não há banco de dados central: os dados exibidos são os mesmos que qualquer pessoa pode verificar rodando o próprio nó.

14 Segurança

Ameaça	Mitigação
Alterar transações confirmadas	Cada bloco compromete transações (Merkle), estado (LtHash) e o bloco anterior; reescrever o histórico exige refazer todo o trabalho acumulado.
Gasto duplo / ataque de 51%	Cadeia de maior trabalho; recompensas com maturação de 100 blocos; reorganizações acima de 720 blocos recusadas; checkpoints. Recomenda-se aguardar 10 confirmações para valores comuns e 60+ para valores altos.
Inflação indevida	Emissão calculada deterministicamente e verificada por todos os nós; teto de 50M checado em cada bloco; aritmética com verificação de overflow.
Replay de transações	Nonce por conta e `chain_id` por rede.
Maleabilidade	Codificação canônica e Ed25519 estrito.
Manipulação de timestamp	Median time past, deriva máxima de 180 s, limites do LWMA.
Negação de serviço	Validação barata antes da cara (alvo, altura, tamanho) e PoW verificada antes de aceitar blocos; pool de órfãos limitado a 16 MB; forks mais antigos que a profundidade máxima descartados sem uso de disco; filas de envio limitadas em bytes com contrapressão; pontuação e banimento de pares; limites da API; mempool limitado com expulsão por taxa.
Eclipse / Sybil na rede	Conexões de saída escolhidas pelo próprio nó, múltiplas seeds, limite por IP, gerenciador de endereços persistente. Alturas anunciadas por pares são tratadas como alegações: a mineração só pausa com progresso real de sincronização, e pares que travam a sincronização são penalizados.
Captura da governança	Duas câmaras independentes, moedas de voto bloqueadas, quórum, depósitos anti-spam, limites rígidos dos parâmetros, supply e emissão imutáveis.
Roubo de chaves	Nós nunca guardam chaves de usuários (só o endereço de recompensa); carteira cifrada com Argon2id + ChaCha20-Poly1305; serviço isolado por systemd.
Corrupção de dados	Transações ACID, compromisso de estado verificado a cada bloco e na inicialização.

Tabela 9: Modelo de ameaças e mitigações.

> Uma rede de prova de trabalho pequena é mais vulnerável a ataques de maioria do que redes estabelecidas. A resposta de longo prazo é crescimento: cada VPS minerando aumenta o custo de um ataque. Os limites de reorganização e os checkpoints são salvaguardas da fase inicial.

15 Desempenho e escalabilidade

Todas as medições abaixo foram feitas com a implementação de referência v0.1 (compilação `release`) num VPS OVH de 2 vCPU Haswell, 3,7 GB de RAM e disco SSD de rede — exatamente o tipo de máquina para a qual a rede foi projetada.

Medição	Resultado
Transferência simples (sem memo)	158 bytes
Cabeçalho de bloco	180 bytes
CoinHash (16 MiB) por tentativa	12,0 ms · 83–87 H/s por thread
Verificação sem estado + Ed25519 estrito	≈ 14 400 transações/s por núcleo
Aplicar bloco de 790 kB com 5 000 transferências (inclui assinaturas)	0,39 s (≈ 12 900 tx/s)
Atualizar o LtHash para 5 002 registros alterados	62 ms
Processar e gravar bloco vazio (validação + banco)	0,39 ms
Montar, validar e gravar bloco com 500 transferências	≈ 104 ms
Memória residente do nó minerando (1 thread)	≈ 42 MB
Disco por bloco vazio em regime (cabeçalho, índices)	≈ 765 bytes (≈ 400 MB/ano)
Disco por transação, nó arquivo com índice de endereços (300 mil tx medidas)	≈ 886 bytes
Disco por transação, nó arquivo sem índice de endereços	≈ 437 bytes

Tabela 10: Desempenho medido da implementação de referência.

15.1 Capacidade

Com o limite padrão de 1 MB por bloco e blocos de 60 s, cabem cerca de 6 300 transferências por minuto — aproximadamente **105 transações por segundo** de capacidade sustentada na camada base, várias vezes a do Bitcoin. Um bloco cheio é validado em menos de meio segundo por um único núcleo modesto, ou seja, menos de 1% do intervalo entre blocos: mesmo com a rede no limite, a validação não compete com a mineração nem torna o nó lento. O limite de tamanho é um parâmetro de governança (250 kB a 8 MB) e pode acompanhar a demanda real e a capacidade dos nós.

15.2 Crescimento dos dados

- **Estado:** proporcional às contas ativas (≈ 60 bytes por conta), não ao histórico.
- **Cabeçalhos:** ≈ 400 MB por ano, sempre mantidos, pois provam a cadeia de trabalho.
- **Corpos de blocos:** só existem quando há transações (blocos vazios não ocupam espaço) e são comprimidos com zstd; nós podados mantêm apenas os mais recentes.
- **Dados de undo:** mantidos apenas para os últimos 736 blocos.
- **Índices:** o índice de endereços é opcional (`storage.address_index`); nós que não servem exploradores podem desativá-lo.

Um nó minerador comum pode rodar podado e sem índice de endereços por muitos anos num disco de 5–10 GB; os nós arquivo (como os seeds e os que alimentam o site) guardam o histórico completo, que permanece público e verificável.

15.3 Sincronização

Um novo nó baixa blocos em lotes de 500, verifica a prova de trabalho em paralelo em todos os núcleos e aplica cada bloco em ordem, conferindo a raiz de estado. O custo dominante é a prova de trabalho (12 ms por bloco por núcleo): um ano de cadeia exige ≈ 1,75 hora de CPU, ou ≈ 53 minutos numa máquina de 2 núcleos. A versão 0.2 adicionará sincronização por **snapshot** de estado, verificável pelo `state_root` dos cabeçalhos, para que novos nós fiquem prontos em minutos.

16 Lançamento da rede e roteiro

16.1 Plano de lançamento

1. Publicar código, binários assinados por checksum, instalador e este whitepaper.
2. Operar uma **testnet** pública para validação por desenvolvedores.
3. Colocar no ar os dois nós seed (`seed1` e `seed2.the-coin.cloud`) e o site em máquina separada.
4. Fixar o timestamp da gênese da mainnet e iniciar a mineração nos nós seed.
5. Abrir o instalador ao público; cada novo nó sincroniza e passa a minerar automaticamente.
6. Após as primeiras semanas, adicionar checkpoints de alturas consolidadas em novas versões.

16.2 Roteiro

- **v0.2** — criptografia do transporte P2P (Noise), sincronização rápida por **snapshot** de estado verificado pelo `state_root`, carteira gráfica e web, provas Merkle para clientes leves.
- **v0.3** — VM WebAssembly com medição de custo para contratos gerais, ativada por governança; novos modelos de pagamento (faturas recorrentes com débito autorizado, pagamentos condicionais a oráculos).
- **v0.4** — canais de pagamento sobre HTLC, integração com exchanges e ferramentas para comércio.
- **Contínuo** — auditorias externas, programa de recompensa por falhas, diversificação de implementações.

17 Conclusão

The Coin combina a robustez comprovada da prova de trabalho e de uma política monetária fixa com escolhas técnicas voltadas a máquinas modestas: uma prova de trabalho memory-hard que valoriza CPUs comuns, validação barata com compromisso de estado homomórfico, armazenamento compacto e um nó que não disputa recursos com o próprio sistema. Sobre essa base, contratos de pagamento seguros e uma governança que exige o consenso de detentores e mineradores permitem que a rede seja útil desde o primeiro dia e evolua de forma democrática. A versão 0.1 é o começo: o código é aberto, os padrões são documentados e qualquer pessoa pode rodar um nó, criar uma carteira ou propor melhorias.

Apêndice A — Parâmetros das redes

Parâmetro	Mainnet	Testnet	Regtest
Magic P2P	`TCN1`	`TCNT`	`TCNR`
chain_id	`0x54430001`	`0x54430002`	`0x54430003`
Portas P2P / API	7333 / 7334	17333 / 17334	27333 / 27334
Prefixo de endereço	`tc`	`tct`	`tcr`
CoinHash	16 MiB, t=1	16 MiB, t=1	64 KiB, t=1
Alvo da gênese / limite	$2^{256}/2^{13}$ / $2^{256}/2^{8}$	$2^{256}/2^{10}$ / $2^{256}/2^{6}$	máximo, sem ajuste
Tempo de bloco / janela LWMA	60 s / 60	60 s / 60	60 s / —
Recompensa inicial	40 TCN	40 TCN	40 TCN
Halving	625 000 blocos	625 000 blocos	150 blocos
Maturação	100	100	5
Reorganização máxima	720	720	720
Bloco máximo (padrão)	1 MB	1 MB	1 MB
Taxa mínima (padrão)	10 motes/byte	10 motes/byte	1 mote/byte

Parâmetro	Mainnet	Testnet	Regtest
Votação / ativação	20 160 / 2 880	1 440 / 720	20 / 5
Quórum / aprovação / mineradores	10% / 66,67% / 60%	5% / 66,67% / 50%	10% / 66,67% / 50%

Apêndice B — Referências

1. S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.
2. A. Biryukov, D. Dinu, D. Khovratovich, S. Josefsson. *RFC 9106: Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications*, 2021.
3. J. O'Connor, J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn. *BLAKE3: one function, fast everywhere*, 2020.
4. S. Josefsson, I. Liusvaara. *RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)*, 2017.
5. M. Bellare, D. Micciancio. *A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost*, EUROCRYPT 1997.
6. K. Lewi, W. Kim, I. Maykov, S. Weis. *Securing Update Propagation with Homomorphic Hashing* (LtHash), 2019.
7. Zawy12. *LWMA difficulty algorithm*, 2018–2019.
8. P. Wuille. *BIP-350: Bech32m format for v1+ witness addresses*, 2020.
9. M. Palatinus et al. *BIP-39: Mnemonic code for generating deterministic keys*, 2013.
10. SatoshiLabs. *SLIP-0010: Universal private key derivation from master private key*, 2016.
11. NEAR Protocol. *Borsh: Binary Object Representation Serializer for Hashing*.